Apple IIGS
#66: ExpressLoad Philosophy

Revised by: Matt Deatherage May 1992
Written by: Matt Deatherage September 1989

This Technical Note discusses the ExpressLoad feature and how it relates to the standard Loader and your application.

Changes since September 1990: Clarified some changes now that ExpressLoad and the System Loader are combined to be "Loader 4.0" in System Software 6.0. Completely removed the note about not calling `Close(0)` since it's not relevant.

Speedy the Loader Helper

ExpressLoad is a GS/OS feature which is usually present with System Software 5.0 (if the `ExpressLoad` file is present and there's more than 512K of RAM), and always on System Software 5.0.4 and later. In fact, ExpressLoad is no longer a separate file in System Software 6.0; it's included in the System Loader version 4.0. Even though ExpressLoad is part of the Loader, we refer to its functionality separately to distinguish how the Loader takes special advantage of "expressed" files.

ExpressLoad operates on Object Module Format (OMF) files which have been "expressed," using either the APW tool Express (or it's MPW counterpart, ExpressIIGS) or created that way by a linker. Expressed files contain a dynamic data segment named either `ExpressLoad` or `~ExpressLoad` at the beginning of the file. (Current versions of Express and ExpressIIGS create `~ExpressLoad` segments, which is the preferred naming convention; older versions created `ExpressLoad` segments, and should be re-Expressed for future compatibility.) This segment contains information which allows the Loader to load these files more quickly, including such things as file offsets to segment headers, mappings of old segment numbers to new segment numbers (these files may have their segments rearranged for optimal performance), and file offsets to relocation dictionaries.

Two Loader Components, Two Missions, One Function

The System Loader's function is to interpret OMF. It takes files on disk (or in memory) and transforms them from load files into relocated 65816 code. It does this very well, but in a very straightforward way. For example, when the System Loader sees the instruction to right-shift a value n times, it loads a register with the value and performs a right-shift n times.

ExpressLoad has a different mission. It relies upon the rest of the System Loader to handle OMF in a straightforward fashion so it can concentrate upon handling the most common OMF cases in the fastest possible way. For example, when asked for a specific segment in a load file, the System Loader "walks" the OMF until it finds the desired segment. ExpressLoad, however, goes directly to the desired segment since an Expressed file contains precalculated offsets to each segment in the `ExpressLoad` segment.

Since ExpressLoad focuses on the common things performed by the majority of applications, it may not support those applications which rely upon certain features of OMF or the System Loader. In these cases, the System Loader loads the file as is expected.

ExpressLoad always gets first crack at loading a file, and if it is an Expressed file that ExpressLoad can handle, it loads it. If the file is not an Expressed file, the regular System Loader loads it instead. ExpressLoad also gets first shot at other loader calls.

Because an Expressed file is a standard OMF file with an additional segment, Expressed files are almost fully compatible with the System Loader (although it cannot load them any faster than before). Refer the following section for potential problems.

Working With ExpressLoad

As ExpressLoad is intimate in its relationship with the System Loader, most applications work seamlessly with it; however, there are some potential problems about which you should be aware.

- Don't mix Expressed files and normal OMF files with the same user ID. For example, if your application uses `InitialLoad` with a separate file, make sure that if it and your main application share the same user ID that they are both either Expressed files or normal OMF files.

- Don't use a user ID of zero. In the past, use of zero told the System Loader to use the current user ID; however, now both the System Loader and ExpressLoad have a current user ID. Be specific about user IDs when loading. This is fixed in 6.0, but is still a good thing to avoid for compatibility with System Software 5.0 through 5.0.4.

- Avoid loading and unloading segments by number. Since Expressed files may have their segments rearranged, if an Expressed file is loaded by the System Loader, references to segments by number may be incorrect.

- Avoid using `GetLoadSegInfo` before System Software 6.0. This call returns System Loader data structures which are not supported by ExpressLoad previous to 6.0. In System Software 6.0 and later, the combined Loaders return correct information for `GetLoadSegInfo` regardless of whether the load file is expressed or not.

- Don't try to load segments in files which have not been loaded with the call `InitialLoad`. This process was never a very good idea, and it is now apt to cause problems.

- Don't have segments that link to other files. ExpressLoad does not support this type of link.

Further Reference

- *GS/OS Reference*